CS388G Problem Set #6

Nidhi Kadkol (nk9368)

May 2019

1

Consider the following instance of the load balancing problem: There are k bins and $2k+1$ items. The first 2 items have weight $k+1$, the next 2 items have weight $k+2$, each of the items in the third pair have weight $k+3$, and so on until the $(k-1)^{th}$ pair of items which each have weight $k+(k-1)$. This leaves us with $(2k+1)-2(k-1)=3$ items. Each of these 3 items have weight k.

Now we process the items according to the second algorithm discussed in class. We place the items in decreasing order of weight into one empty bin at a time. So after k steps, we have placed one item in each bin and the weights of each consecutive pair of bins is $k+(k-1)$, $k+(k-2)$, and so on. If k is even, the last 2 bins have weight $k+k/2$, otherwise only the last bin has weight $k+\lfloor k/2 \rfloor$.

Now we process the next k items in decreasing order of weight. If k is odd, the first item we process is $k+\lfloor k/2 \rfloor$ and we add to the bin with weight $k+\lfloor k/2 \rfloor$, so the updated bin weight is $3k-1$. After that, we process the 2 items with weights $k+\lfloor k/2 \rfloor - 1$ and add them each to the 2 bins with weights $k+\lfloor k/2 \rfloor + 1$, so those 2 bins have an updated weight of $3k-1$. We continue this until we add the 2 items with weights k respectively to the bins with weights $k+k-1$ to give us updated bin weights of $3k-1$. If k is even, then the first 2 items we process are $k+k/2-1$ and we add them to the bins with weights $k+k/2$, giving us updated bin weights of $3k-1$. We continue in this way and thus at the end of the k items all the bins have a weight of $3k-1$ irrespective of whether k is odd or even.

Now we have one more item left to process, which is of weight k. It will be added to any bin, giving us a max bin weight of $4k-1$.

However, it is possible to achieve a max bin load of $3k$, by putting the 3 items with weight k in one bin and putting the heaviest and lightest item in bin 2, the next heaviest and lightest item in bin 3, and so on. So the first bin has a load of $3k$, and all the other items also have a load of $3k$ ($(k+1)+(k+(k-1)$ in bin 2, $(k+2)+(k+k-2)$ in bin 3, and so on).

Thus, the approximation ratio is $(4k-1)/3k$. By choosing k sufficiently large, the approximation ratio exceeds $4/3-\epsilon$ for $\epsilon > 0$.

2(a)

Claim 1. The size of a maximum clique in $G^{(k)}$ is at least m^k where m is the size of a maximum clique in G.

Proof. Suppose C is a clique of m vertices in G. Let $C^{(k)}$ denote the set of all k-tuples of vertices in C. Then there will be an edge from $E^{(k)}$ between every pair of vertices in $C^{(k)}$. This is because in C, every vertex is adjacent to every other vertex. Hence, every element of a k-tuple containing vertices in C will be either adjacent or equal to the corresponding element of any other k-tuple of vertices in C. Thus, $C^{(k)}$ is a clique in $G^{(k)}$. There are a total of m^k k-tuples of vertices in C. So the size of $C^{(k)}$ is m^k. This is for any arbitrary clique C. C could also be a maximum clique. In that case, m is the size of a maximum clique in G and there exists a clique of size m^k in $G^{(k)}$. Thus, the size of a maximum clique in $G^{(k)}$ is at least m^k. ■

Now we show that the size of a maximum clique in $G^{(k)}$ cannot be more than m^k, where m is the size of a maximum clique in G.

Proof. We proceed by induction on k.

Base Case: When $k = 1$, $V^{(1)} = V$ and $E^{(1)} = E$. So the size of a maximum clique in $G^{(1)}$ = size of a maximum clique in $G = m = m^1$.

Induction Hypothesis: For $k \geq 1$, the size of a maximum clique in $G^{(k)} = m^k$.

Induction Step: Now consider $G^{(k+1)}$. Let u be a $(k+1)$-tuple of vertices. Let u' be the corresponding k-tuple of vertices which has the first k entries of u. It is clear that if there is an edge belonging to $E^{(k+1)}$ between u and v, then there is an edge belonging to $E^{(k)}$ between u' and v'.

Let us suppose, if possible that the size of a maximum clique C in $G^{(k+1)}$ is equal to $n > m^{k+1}$. Construct $C' = \{u' | u \in C\}$. So now C' is also a clique, and it is a clique in $G^{(k)}$. Now we calculate how many vertices are in C'. After condensing each vertex u in C to u', we remove duplicate vertices. The number of duplicate vertices would be the number of vertices in C which have the first k entries in their tuples the same, and differ only in the last entry. The maximum number of vertices would be removed when we can split C into l disjoint groups, each group having m vertices which differ only in their last entry and $n = lm$. In this case, C' would take one condensed k-tuple from each group. Thus, in the worst case, $|C'| = n/m$. $|C'|$ cannot be lower than this, i.e. $|C'| \geq n/m > m^{k+1}/m \implies |C'| > m^k$. However from the induction hypothesis $|C'| \leq m^k$. This is a contradiction. Thus, our assumption that $n > m^{k+1}$ is wrong.

Thus, $n \leq m^{k+1}$. But from claim 1, $n \geq m^{k+1}$. Thus, $n = m^{k+1}$. ■

Hence, the size of a maximum clique in $G^{(k)}$ is m^k where m is the size of a maximum clique in G.

2(b)

We are given that there is a polynomial-time approximation algorithm with constant approximation ratio c for finding a maximum-size clique. We must show that there is a polynomial-time approximation scheme for finding a maximum-size clique, that is, there is a polynomial-time approximation algorithm with approximation ratio $1+\epsilon$ for any $\epsilon > 0$ given to us.

To find a solution to the maximum-size clique problem for a graph G, we construct $G^{(k)}$ and find an approximate solution to the problem in $G^{(k)}$ using the algorithm with constant approximation ratio. If the size of the clique we find is n, and the size of the clique which is the optimal solution to the problem is m, then $n \geq m/c$. Thus, $n^{1/k} \geq m^{1/k}/c^{1/k}$.

$n^{1/k}$ is our solution to the problem for G, and $m^{1/k}$ is the optimal solution to the problem for G. We have the freedom to choose k based on ϵ such that this process gives us a polynomial-time approximation scheme. Hence,

$$c^{1/k} < 1+\epsilon \implies \frac{1}{k} < \log_c(1+\epsilon) \implies k > \frac{1}{\log_c(1+\epsilon)}$$

Since converting G to $G^{(k)}$ can be done in polynomial time in the size of G, we can choose $k > 1/\log_c(1+\epsilon)$ and use the constant ratio approximation algorithm which runs in polynomial time to get a clique of size n, from which we get our clique of size $n^{1/k}$ which is at most (the optimal size of the clique in G)/$(1+\epsilon)$. Hence, we have a polynomial-time approximation scheme for the problem.

3

Let us choose $p = 1/2$ and $l = a \log n$ where a is sufficiently large constant, that is $a \geq 100n$.

We first have to show that with these values a feasible solution exists, that is, the family $\mathcal{F}$ of sets is a set cover for S. Let $\mathcal{F} = \{T_1, T_2, \cdots, T_l\}$ and e be an element of S. Then

$$\begin{aligned}
P(e \in \mathcal{F}) &= P(e \in \text{(at least one of the sets in } \mathcal{F})) \\
&= 1 - P(e \notin \text{any set in } \mathcal{F}) \\
&= 1 - P(e \notin T_1) \times P(e \notin T_2) \times \cdots (e \notin T_l) \\
&= 1 - (1-p)^l \\
&= 1 - (1/2)^{a \log n} \\
&= 1 - (2^{a \log n})^{-1} \\
&= 1 - n^{-a} \\
&= 1 - 1/n^a
\end{aligned}$$

Now, we shall show that $\mathcal{F}$ is a set cover with high probability.

$$\begin{aligned} P(\mathcal{F} \text{ is a cover}) &= \prod_{e \in S} P(e \in \mathcal{F}) \\ &= \prod_{i=1}^{n} (1 - 1/n^a) \\ &= (1 - 1/n^a)^n \end{aligned}$$

This is very close to 1. Thus, a feasible solution exists with very high probability.

Now we consider the fractional objective function $\sum_{T \in \mathcal{F}} x_T$ where $x_T \geq 0$ for each set in the fractional optimal set cover. We also have the constraint that for each element e in S, $\sum_{T \in \mathcal{F}: e \in T} x_T \geq 1$.

Let X_e denote the number of sets that an arbitrary element e belongs to in the optimal fractional cover. Since the probability p that e belongs to each set is independent of it belonging to other sets, X_e is a binomial random variable with $(n, p) = (l, 1/2)$. From Chernoff bounds, we have that

$$P(X_e \leq (1 - \delta) l/2) \leq \exp\left(\frac{-\delta^2 l}{2}\right) \text{ for all } \delta \in [0, 1]$$

That is, $X_e < l/2$ with very low probability. We set each x_T in our relaxed LP to be c/l. Then our constraint becomes $\sum_{T \in \mathcal{F}: e \in T} c/l = \frac{c}{l} X_e >= \frac{c}{l}\frac{l}{2} = c/2$ with high probability. Furthermore, the value of the objective function is $\sum_{T \in \mathcal{F}} c/l = c$ which is $O(1)$. Thus, if each x_T has a value greater than or equal to $2/l$, i.e. $c \geq 2$, then the constraints are satisfied with high probability and the objective function value is a constant. This means that the optimal fractional objective which is less than or equal to this objective function value is $O(1)$.

Now we consider the integral objective function. We want to show that the number of sets required is of the order of $\Omega(\log n)$. We will prove this by contradiction. Suppose that fewer than $\log n$ sets are chosen, say $(\log n)/k$. Let $F \subseteq \mathcal{F}$ denote this set of sets. Then the probability that an element e belongs to any of the sets in F is $1 - P(e \notin$ any of these sets$) = 1 - (1 - 1/2)^{(\log n)/k} = 1 - \frac{1}{n^{1/k}}$. Thus the probability that these sets form a cover $= \left(1 - \frac{1}{n^{1/k}}\right)^n$. Using Bernoulli's inequality that $(1 + x)^r \leq e^{rx}$ for any real numbers x and $r > 0$, we get that the probability that F is a cover $\leq \exp(-n^{1-1/k})$. The number of such sets F is $\binom{l}{(\log n)/k} = \binom{a \log n}{(\log n)/k} < (\log n)^{\log n}$. Thus, there are a polynomial number of such sets F and each of them has an exponentially small probability of being a set cover. Thus, to be able to cover the elements with a high probability, we need $\Omega(\log n)$ sets.

Thus, with our values of $p = 1/2$ and $l = a \log n$, there is a positive probability that the optimal fractional objective function value is $O(1)$, and the optimal integral objective function value is $\Omega(\log n)$.

4

Suppose $d = 10k + 1$, and consider the transpose permutation that routes a packet from a source $a_1a_2 \cdots a_{5k}a_{5k+1}a_{5k+2}a_{5k+3} \cdots a_{10k+1}$ to destination $a_{5k+2}a_{5k+3} \cdots a_{10k+1}\overline{a_{5k+1}}a_1a_2 \cdots a_{5k}$ where a_{5k+1} changes the bit from 1 to 0 and vice versa.

Now consider the set of all sources such that $a_{5k+1} = 1$, $a_{5k+2}a_{5k+3} \cdots a_{10k+1} = 00 \cdots 0$, and $a_1a_2 \cdots a_{5k}$ can be anything.

We will count the number of packets from sources in this set that travel along the dimension $5k + 1$ edge E from $00 \cdots 000 \cdots 0$ to $00 \cdots 0100 \cdots 0$. We do so by splitting X into disjoint sets $X_0, X_1, \cdots, X_{5k}$ where X_i denotes the set of all sources in X that have i 1s in the first k bit positions.

The number of edges that a packet from a source in X_i needs to travel is equal to the number of incorrect bit positions = (i 1s to be corrected to 0s in the first half) + (1 for the $(5k+1)^{th}$ bit to be corrected from 1 to 0) + (i 0s to be corrected to 1s in the second half) = $2i + 1$. Thus, the total number of ways that a packet from X_i can travel from source to destination is $(2i + 1)!$.

Now to count the number of ways that a packet from a source in X_i travels from its source to its destination along edge E, we see that it travels along E only if the i 1s in the first half are corrected to 0, and then the next bit that is corrected is bit $(5k + 1)$, followed by the correcting the 0s to 1s in the second half. Thus the number of ways the packet travels along E is (the number of ways to correct the i 1s to 0s in the first half) $\times$ (the number of ways to correct the i 0s to 1s in the second half) $= i!\, i!$

Thus, the probability of a packet from a source in X_i travelling along E is $p_i = \frac{i!i!}{(2i+1)!}$. Hence, the expected number of packets in X_i that use E is $|X_i|p_i$.

From linearity of expectation, he expected number of packets from X that use edge E when they move from their source in X to their destination is

$$
\begin{aligned}
&= \sum_{i=0}^{5k} |X_i| p_i \\
&= \sum_{i=0}^{5k} \binom{5k}{i} \frac{i!\, i!}{(2i+1)!} \\
&= \sum_{i=0}^{5k} \frac{5k!}{i!(5k-i)!} \frac{i!\, i!}{(2i+1)!} \\
&\geq \frac{5k!}{k!(5k-k)!} \frac{k!\, k!}{(2k+1)!} \text{ (since each term in the sum is positive so the sum is greater than each term)} \\
&= \frac{5k!\, k!}{4k!(2k+1)!} \\
&= \frac{5k(5k-1)\dots(4k+1)}{(2k+1)(2k)\dots(k+1)} \\
&= \frac{1}{2k+1} \frac{5k}{2k} \frac{5k-1}{2k-1} \dots \frac{4k+1}{k+1} \\
&= \frac{1}{2k+1} \frac{5}{2} \frac{5k-1}{2k-1} \dots \frac{4k+1}{k+1} \text{ (since } k > 0) \\
&> \frac{1}{2k+1} \left(\frac{5}{2}\right)^k \\
&> \frac{1}{2k+1} (2)^k
\end{aligned}
$$

Since k is of the order $\log n$, we get that the expected number of packets is at least greater than a polynomial in n. Thus, the expected time to route this permutation is $\Omega(n^\epsilon)$ for some $\epsilon > 0$.